PCI-1002

Software Manual
[for Windows 95/98/NT/2000/XP]

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damage consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, ICP DAS assumes no responsibility for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 2000 by ICP DAS. All rights are reserved.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

Table of Contents

1. Introduction 3
1.1 Software Installation 4
1.2 References 5
2. Declaration Files 6
2.1 P100X.H 7
2.2 P100Xu.C 9
2.3 P100X.BAS 10
2.4 P100Xu.BAS 12
2.5 P100X.PAS 13
2.6 P100Xu.PAS 16
3. Demo Result 17
3.1 Visual C++ 17
3.2 Visual Basic 18
3.3 Delphi 19
3.4 Borland C++ Builder 20
4. Description of Functions 21
4.1 The Configuration Code Table 23
4.2 The Test Functions 24
4.2.1 P100X_FloatSub2 24
4.2.2 P100X_ShortSub2 24
4.2.3 P100X_GetDllVersion 25
4.2.4 P100X_GetDriverVersion 25
4.3 The DI/O Functions 26
4.3.1 P100X_Di 26
4.3.2 P100X_Do 26
4.4 The AD Fixed-mode Functions 27
4.4.1 P100X_SetChannelConfig 27
4.4.2 P100X_Polling 28
4.4.3 P100X_AdPolling 29
4.4.4 P100X_AdsPolling 30
4.4.5 P100X_AdsPacer 31
4.5 The Driver Functions 32
4.5.1 P100X_DriverInit 32
4.5.2 P100X_DriverClose 32
4.5.3 P100X_GetConfigAddressSpace 33
4.5.4 P100X_WhichBoardActive 33
4.5.5 P100X_ActiveBoard 34
4.5.6 P100X_GetIrqNo 34
4.6 The Interrupt Functions 35
4.6.1 P100X_InstallIrq 35
4.6.2 P100X_GetIntCount 35
4.6.3 P100X_INT_AdStart 36
4.6.4 P100X_ExtInt_AdStart 36
4.6.5 P100X_INT_AdStop 37
4.6.6 P100X_GetBuffer 37
4.6.7 P100X_GetFloatBuffer 38
4.6.8 Architecture of Interrupt mode 39
5. Program Architecture 43
6. Problems Report 44

1. Introduction

The **PCI-1002 Toolkit** is a collection of DLLs and device-driver for Windows 95/98/NT 4.0 and Windows 2000 application. These DLLs are 32 bits and can be called by Visual C++, BC++, Visual BASIC, Delphi and LabVIEW.

The PCI-1002 Toolkit consists of these DLLs and device driver:

- P100X.DLL, P100X.LIB →for PCI-1002 card
- P100X.VXD →PCI-1002 Device driver for Windows 95/98
- P100X.SYS →PCI-1002 Device driver for Windows NT

These DLLs can perform a variety of data acquisition operations as follows:

- Get software version
- Initialization
- Digital Input/Output
- A/D conversion

1.1 Software Installation

A). The CD Disc Contents:

```
|--\README.TXT
|
|--\Win95
|       |--\README.TXT
|       |--\SETUP          ← sub-directory
|               |--\Setup.exe ← the Setup program for WINDOWS 95/98
|
|--\WinNT
|       |--\README.TXT
|       |--\SETUP          ← sub-directory
|               |--\Setup.exe ← the Setup program for Windows NT
|
|--\Win2000
        |--\README.TXT
        |--\SETUP          ← sub-directory
                |--\Setup.exe ← the Setup program for Windows 2000
```

B). Installation Steps

Step 1). Insert 'PCI-1002 Setup CD' into CD-ROM.

Step 2). Clicking Start/Run in the task Bar

Step 3). Enter the path as:

E:\Win95\SETUP\SETUP.EXE
(For Windows 95 user and the CD-ROM drive is E:)

E:\WinNT\SETUP\SETUP.EXE
(For Windows NT user and the CD-ROM drive is E:)

E:\Win2000\SETUP\SETUP.EXE
(For Windows 2000 user and the CD-ROM drive is E:)

Step 4). Following those instructions in installation process to complete it.

After installed,

- for WINDOWS 95/98 User
  The P100X.DLL and P100X.VXD will be copied into C:\Windows\SYSTEM

- for WINDOWS NT and 2000 User
  The P100X.DLL will be copied into C:\WINNT\SYSTEM32.
  And, the P100X.SYS, NAPWnt.sys will be copied into C:\WINNT\SYSTEM32\DRIVERS.

1.2 References

Please refer to the following user manuals:

- **SoftInst.pdf:**
  To install the software package under Windows 95/98/NT.

- **CallDll.pdf:**
  To call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3.

- **ResCheck.pdf:**
  To check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/NT.

2. Declaration Files

Please refer to user manual "CallDLL.pdf".

```
|--\Driver                    ← some device driver
|       |
|       |--\VB                ← for Visual Basic
|       |       |--\P100X.BAS     ← Declaration file for Visual Basic
|       |       |--\P100Xu.BAS    ← Some function for Visual Basic
|       |
|       |--\VC                ← for Visual C++
|       |       |--\P100X.H       ← Header file
|       |       |--\P100X.LIB     ← Import library for VC only
|       |
|       |--\Delphi            ← for Delphi
|       |       |--\P100X.PAS     ← Declaration file
|       |       |--\P100Xu.PAS    ← Some function for Delphi
|       |
|       |--\BCB               ← for Borland C++ Builder 3.0
|               |--\P100X.H       ← Header file
|               |--\P100Xu.C      ← Some function for BCB
|               |--\P100X.LIB     ← Import library file for BCB only
```

2.1 P100X.H

```
#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

// return code
#define P100X_NoError                0
#define P100X_DriverHandleError      1
#define P100X_DriverCallError        2
#define P100X_AdControllerError      3
#define P100X_ConfigCodeError        4
#define P100X_DriverNoOpen           5
#define P100X_AdPollingTimeOut       6
#define P100X_FindBoardError         7
#define P100X_AdChannelError         8
#define P100X_DaChannelError         9
#define P100X_InvalidateDelay        10
#define P100X_DelayTimeOut           11
#define P100X_InvalidateData         12
#define P100X_TimeoutError           13
#define P100X_ExceedBoardNumber      14
#define P100X_NotFoundBoard          15
#define P100X_OpenError              16
#define P100X_FindTwoBoardError      17
#define P100X_GetIntCountError       18
#define P100X_InstallIrqError        19
#define P100X_AllocateMemoryError    20

EXPORTS float      CALLBACK P100X_FloatSub(float fA, float fB);
EXPORTS short      CALLBACK P100X_ShortSub(short nA, short nB);
EXPORTS WORD   CALLBACK P100X_GetDllVersion(void);

EXPORTS WORD   CALLBACK P100X_DriverInit(WORD *wTotalBoards);
EXPORTS void       CALLBACK P100X_DriverClose(void);
EXPORTS WORD   CALLBACK P100X_GetDriverVersion
        (WORD *wDriverVersion);
EXPORTS WORD   CALLBACK P100X_GetIrqNo( WORD  *IrqNo);

EXPORTS WORD   CALLBACK P100X_GetConfigAddressSpace
        (WORD wBoardNo, WORD *wAddress0,
         WORD *wAddress1, WORD *wAddress2);

EXPORTS WORD   CALLBACK P100X_ActiveBoard( WORD wBoardNo );
EXPORTS WORD   CALLBACK P100X_WhichBoardActive(void);
```

```
EXPORTS void       CALLBACK P100X_SetupTimer
        (WORD wChannel, WORD wCoef);
EXPORTS WORD   CALLBACK P100X_Delay(WORD wDownCount);

EXPORTS void       CALLBACK P100X_Do(WORD wOutData);
EXPORTS WORD   CALLBACK P100X_Di(WORD *wDiData);

EXPORTS WORD   CALLBACK P100X_SetChannelConfig
        (WORD wAdChannel, WORD wConfig);
EXPORTS WORD   CALLBACK P100X_Polling(WORD *wAdVal);
EXPORTS WORD   CALLBACK P100X_AdPolling(float *fAdVal);
EXPORTS WORD   CALLBACK P100X_AdsPolling
        (float fAdVal[], WORD wNum);

EXPORTS WORD   CALLBACK P100X_AdsPacer
        (float fAdVal[], WORD wNum,  WORD wSamplingDiv);


EXPORTS WORD   CALLBACK P100X_InstallIrq
        (HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD   CALLBACK P100X_GetBuffer
        (DWORD dwNum, WORD wBuf[]);
EXPORTS WORD   CALLBACK P100X_GetFloatBuffer
        (DWORD dwNum, float fAdVal[]);

EXPORTS WORD   CALLBACK P100X_GetIntCount(DWORD *dwVal);
EXPORTS WORD   CALLBACK P100X_INT_AdStart
        (WORD Ch, WORD Gain, WORD wFreqDiv);
EXPORTS WORD   CALLBACK P100X_ExtINT_AdStart
(WORD Ch, WORD Gain);

EXPORTS WORD   CALLBACK P100X_INT_AdStop();
```

2.2 P100Xu.C

```
#include <math.h>


//*-----------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                       *
//* HiLo : 1 --> High Gain , 0 --> Low Gain             *
//* Gain : 0-3                                          *
//*-----------------------------------------------------*
float P100X_AD2F(Word hex, int HiLo,int Gain )
{
   float ZeroBase, VoltageRange, FullRange ;

   ZeroBase  = 2048.0 ;
   FullRange = 2048.0 ;
   VoltageRange = 10.0 ;
   Gain = Gain % 16 ;
   if ( (Gain < 0) || (Gain > 3) )
      return -100.0;

   if ( HiLo == 0 )   //Low-Gain
      return ((((hex - ZeroBase) / FullRange) * VoltageRange) / pow(  2 , Gain));
   else
      return ((((hex - ZeroBase) / FullRange) * VoltageRange) / pow( 10 , Gain));

}
```

2.3 P100X.BAS

```
Attribute VB_Name = "P100X"

' return code
Global Const P100X_NoError                = 0
Global Const P100X_DriverHandleError      = 1
Global Const P100X_DriverCallError        = 2
Global Const P100X_AdControllerError      = 3
Global Const P100X_ConfigCodeError        = 4
Global Const P100X_DriverNoOpen           = 5
Global Const P100X_AdPollingTimeOut       = 6
Global Const P100X_FindBoardError         = 7
Global Const P100X_AdChannelError         = 8
Global Const P100X_DaChannelError         = 9
Global Const P100X_InvalidateDelay        = 10
Global Const P100X_DelayTimeOut           = 11
Global Const P100X_InvalidateData         = 12
Global Const P100X_TimeoutError           = 13
Global Const P100X_ExceedBoardNumber      = 14
Global Const P100X_NotFoundBoard          = 15
Global Const P100X_OpenError              = 16
Global Const P100X_FindTwoBoardError      = 17
Global Const P100X_GetIntCountError       = 18
Global Const P100X_InstallIrqError        = 19
Global Const P100X_AllocateMemoryError    = 20

' Function of Test
Declare Function P100X_FloatSub Lib "P100X.DLL" _
        (ByVal fA As Single, ByVal fB As Single) As Single
Declare Function P100X_ShortSub Lib "P100X.DLL" _
        (ByVal nA As Integer, ByVal nB As Integer) As Integer
Declare Function P100X_GetDllVersion Lib "P100X.DLL" () As Integer

' Function of Driver
Declare Function P100X_DriverInit Lib "P100X.DLL" _
        (wTotalBoards As Integer) As Integer
Declare Sub P100X_DriverClose Lib "P100X.DLL" ()
Declare Function P100X_GetDriverVersion Lib "P100X.DLL" _
        (wDriverVersion As Integer) As Integer
Declare Function P100X_GetIrqNo Lib "P100X.DLL" (IrqNo As Integer) _
        As Integer
Declare Function P100X_GetConfigAddressSpace Lib "P100X.DLL" _
        (ByVal wBoardNo As Integer, wAddrTimer As Integer, _
        wAddrDio As Integer, wAddrAd As Integer) As Integer
Declare Function P100X_ActiveBoard Lib "P100X.DLL" _
        (ByVal wBoardNo As Integer) As Integer
Declare Function P100X_WhichBoardActive Lib "P100X.DLL" () As Integer
```

```
Declare Sub P100X_SetupTimer Lib "P100X.DLL" _
        (ByVal wChannel As Integer, ByVal wCoef As Integer)
Declare Function P100X_Delay Lib "P100X.DLL" _
        (ByVal wDownCount As Integer) As Integer

' Function of DI/DO
Declare Sub P100X_Do Lib "P100X.DLL" (ByVal wOutData As Integer)
Declare Function P100X_Di Lib "P100X.DLL" (wDiData As Integer) As Integer

' Function of AD
Declare Function P100X_SetChannelConfig Lib "P100X.DLL" _
        (ByVal wAdChannel As Integer, ByVal wConfig As Integer) As Integer
Declare Function P100X_Polling Lib "P100X.DLL" _
        (wAdVal As Integer) As Integer
Declare Function P100X_AdPolling Lib "P100X.DLL" _
        (fAdVal As Single) As Integer
Declare Function P100X_AdsPolling Lib "P100X.DLL" _
        (fAdVal As Single, ByVal wNum As Integer) As Integer
Declare Function P100X_AdsPacer Lib "P100X.DLL" (fAdVal As Single, _
        ByVal wNum As Integer, ByVal wSamplingDiv As Integer) As Integer

' Function of Interrupt
Declare Function P100X_InstallIrq Lib "P100X.DLL" _
        (hEvent As Long, ByVal dwCount As Long) As Integer
Declare Function P100X_GetBuffer Lib "P100X.DLL" _
        (ByVal dwNum As Long, wBuf As Integer) As Integer
Declare Function P100X_GetFloatBuffer Lib "P100X.DLL" _
        (ByVal dwNum As Long, fAdVal As Single) As Integer
Declare Function P100X_INT_AdStart Lib "P100X.DLL" _
        (ByVal Ch As Integer, ByVal Gain As Integer, _
        ByVal wFreqDiv As Integer) As Integer
Declare Function P100X_ExtINT_AdStart Lib "P100X.DLL" _
        (ByVal Ch As Integer, ByVal Gain As Integer) As Integer

Declare Function P100X_INT_AdStop Lib "P100X.DLL" () As Integer
Declare Function P100X_GetIntCount Lib "P100X.DLL" (dwVal As Long) _
        As Integer
```

2.4 P100Xu.BAS

```
'*-----------------------------------------------------*
'* Return voltage value or -100.0 if any error occurs  *
'* or parameter is out of range.                       *
'* HiLo : 1 --> High Gain , 0 --> Low Gain             *
'* Gain : 0-3                                          *
'*-----------------------------------------------------*
Function P100X_AD2F(ByVal hex, HiLo, Gain As Integer) As Single
   Dim ZeroBase, BullRange, VoltageRange As Single

   ZeroBase = 2048#
   FullRange = 2048#
   VoltageRange = 10#
   Gain = Gain Mod 16

   If Gain < 0 Or Gain > 3 Then
      P100X_AD2F = -100#
      Exit Function
   End If

   If HiLo = 0 Then   'Low-Gain
      P100X_AD2F = ((((hex - ZeroBase) / FullRange) * VoltageRange) / (2 ^
Gain))
   Else
      P100X_AD2F = ((((hex - ZeroBase) / FullRange) * VoltageRange) / (10 ^
Gain))
   End If
End Function
```

2.5 P100X.PAS

```
unit P100X;

interface

type PSingle=^Single;
type PWord=^Word;

const
// return code
 P100X_NoError                =   0;
 P100X_DriverHandleError      =   1;
 P100X_DriverCallError        =   2;
 P100X_AdControllerError      =   3;
 P100X_ConfigCodeError        =   4;
 P100X_DriverNoOpen           =   5;
 P100X_AdPollingTimeOut       =   6;
 P100X_FindBoardError         =   7;
 P100X_AdChannelError         =   8;
 P100X_DaChannelError         =   9;
 P100X_InvalidateDelay        =   10;
 P100X_DelayTimeOut           =   11;
 P100X_InvalidateData         =   12;
 P100X_TimeoutError           =   13;
 P100X_ExceedBoardNumber      =   14;
 P100X_NotFoundBoard          =   15;
 P100X_OpenError              =   16;
 P100X_FindTwoBoardError      =   17;
 P100X_GetIntCountError       =   18;
 P100X_InstallIrqError        =   19;
 P100X_AllocateMemoryError    =   20;

// Function of Test
function  P100X_FloatSub(fA:Single; fB:Single):Single ; stdCall;
function  P100X_ShortSub(nA:SmallInt; nB:SmallInt):SmallInt ; stdCall;
function  P100X_GetDllVersion:WORD  ; stdCall;
```

```
// Function of Driver
function  P100X_DriverInit(Var wTotalBoards:Word):WORD ; stdCall;
procedure P100X_DriverClose; stdCall;
function  P100X_GetDriverVersion(var wDriverVersion:Word):WORD ; stdCall;
function  P100X_GetIrqNo(Var IrqNo:WORD):WORD; StdCall;
function  P100X_GetConfigAddressSpace(wBoardNo:Word;
        var wAddrTimer:Word; var wAddrDio:Word;
        var wAddrAd:Word) :WORD ; stdCall;
function  P100X_ActiveBoard(wBoardNo:Word):WORD ; stdCall;
function  P100X_WhichBoardActive:WORD ; stdCall;
procedure P100X_SetupTimer(wChannel:Word; wCoef:Word); stdCall;
function  P100X_Delay(wDownCount:Word):Word; StdCall;

// Function of DI/DO
procedure P100X_Do(wOutData:Word); stdCall;
function  P100X_Di(var wDiData:Word):WORD ; stdCall;

// Function of AD
function  P100X_SetChannelConfig
        (wAdChannel:Word; wConfig:Word):WORD ; stdCall;
function  P100X_Polling(var wAdVal:Word):WORD ; stdCall;
function  P100X_AdPolling(var fAdVal:Single):WORD ; stdCall;
function  P100X_AdsPolling(fAdVal:PSingle; wNum:Word):WORD ; stdCall;
function  P100X_AdsPacer(fAdVal:PSingle; wNum:Word;
         wSamplingDiv:Word ):WORD ; stdCall;

// Function of Interrupt
function  P100X_InstallIrq
        (Var hEvent:LongInt; dwCount: LongInt):WORD ; stdCall;
function  P100X_GetBuffer(dwNum:LongInt;wBuf:PWord):WORD ; stdCall;
function  P100X_GetFloatBuffer
        (dwNum:LongInt; fAdVal:PSingle):Word; StdCall;
function  P100X_INT_AdStart
        (Ch:WORD; Gain:WORD; wFreqDiv:Word):WORD ; stdCall;
function  P100X_ExtINT_AdStart
        (Ch:WORD; Gain:WORD):WORD ; stdCall;

function  P100X_INT_AdStop:WORD ; stdCall;
function  P100X_GetIntCount(var dwVal:LongInt):WORD ; stdCall;


implementation

function        100X_FloatSub;    external 'P100X.DLL' name 'P100X_FloatSub';
function        100X_ShortSub;   external 'P100X.DLL' name 'P100X_ShortSub';
function        100X_GetDllVersion;
                external 'P100X.DLL' name 'P100X_GetDllVersion';
function        100X_GetDriverVersion;
                external 'P100X.DLL' name 'P100X_GetDriverVersion';
```

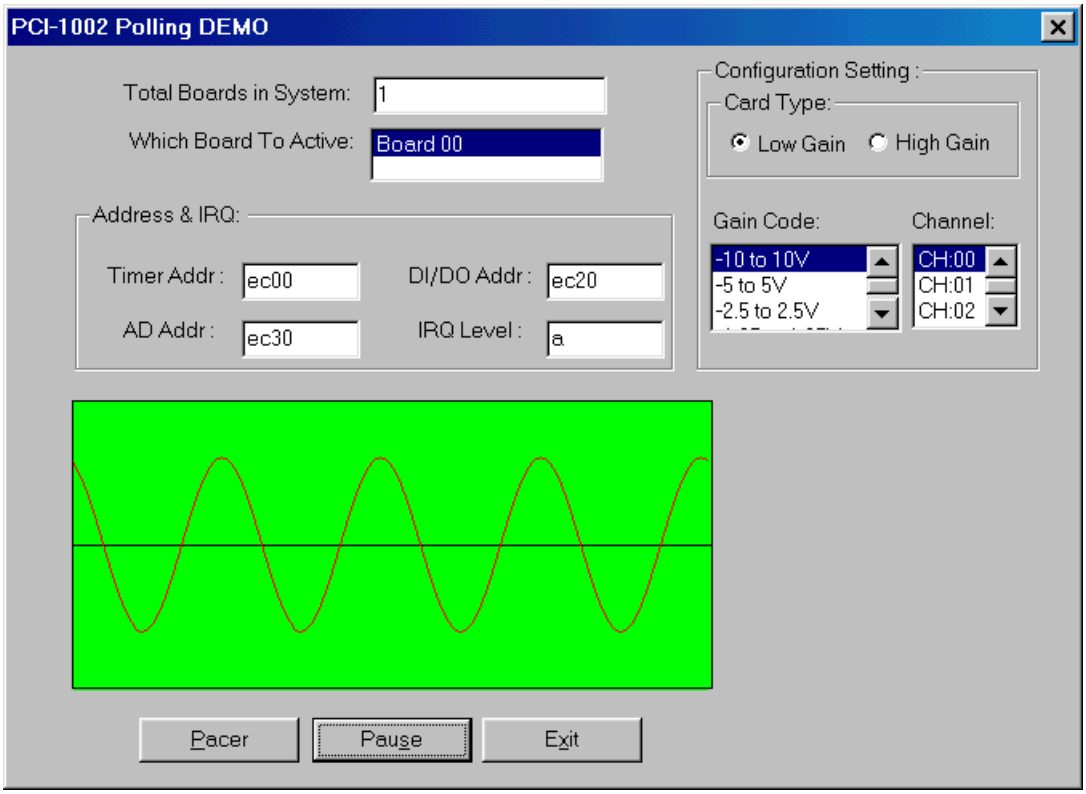
```
function    100X_DriverInit;  external 'P100X.DLL' name 'P100X_DriverInit';
procedure   100X_DriverClose;
            external 'P100X.DLL'        name 'P100X_DriverClose';
function    100X_GetIrqNo;  external 'P100X.DLL' name 'P100X_GetIrqNo';
function    100X_GetConfigAddressSpace;
            external 'P100X.DLL' name 'P100X_GetConfigAddressSpace';
function    100X_ActiveBoard;
            external 'P100X.DLL'        name 'P100X_ActiveBoard';
function    100X_WhichBoardActive;
            external 'P100X.DLL'        name 'P100X_WhichBoardActive';
procedure   100X_SetupTimer;
            external 'P100X.DLL'        name 'P100X_SetupTimer';
function    100X_Delay;     external 'P100X.DLL' name 'P100X_Delay';


procedure   P100X_Do;       external 'P100X.DLL' name 'P100X_Do';
function    P100X_Di;       external 'P100X.DLL' name 'P100X_Di';


function    P100X_SetChannelConfig;
            external 'P100X.DLL'        name 'P100X_SetChannelConfig';
function    P100X_Polling;  external 'P100X.DLL' name 'P100X_Polling';
function    P100X_AdPolling;
            external 'P100X.DLL'        name 'P100X_AdPolling';
function    P100X_AdsPolling;
            external 'P100X.DLL'        name 'P100X_AdsPolling';

function    P100X_AdsPacer;
            external 'P100X.DLL'        name 'P100X_AdsPacer';

function    P100X_InstallIrq;
            external 'P100X.DLL'        name 'P100X_InstallIrq';
function    P100X_INT_AdStart;
            external 'P100X.DLL'        name 'P100X_INT_AdStart';
function    P100X_ExtNT_AdStart;
            external 'P100X.DLL'        name 'P100X_ExtINT_AdStart';

function    P100X_INT_AdStop;
            external 'P100X.DLL'        name 'P100X_INT_AdStop';
function    P100X_GetIntCount;
            external 'P100X.DLL'        name 'P100X_GetIntCount';
function    P100X_GetBuffer;
            external 'P100X.DLL'        name 'P100X_GetBuffer';
function    P100X_GetFloatBuffer;
            external 'P100X.DLL'        name 'P100X_GetFloatBuffer';

end.
```

2.6 P100Xu.PAS

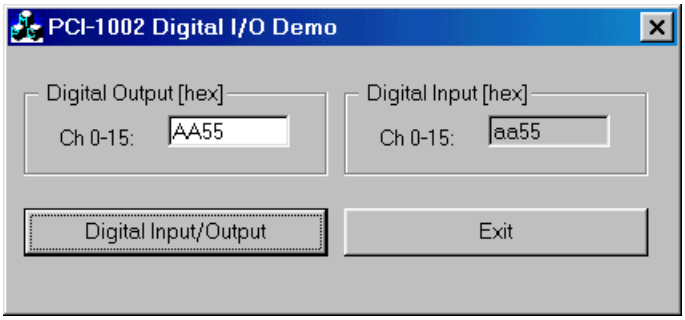
```
unit P100Xu;

interface


Function  P100X_AD2F(hex, HiLo, Gain :Word): Single ; StdCall;


implementation

uses math;


//*-----------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                       *
//* HiLo : 1 --> High Gain , 0 --> Low Gain             *
//* Gain : 0-3                                          *
//*-----------------------------------------------------*
Function  P100X_AD2F(hex, HiLo, Gain :Word): Single ;
Var
   ZeroBase, VoltageRange, FullRange : Single ;

Begin

   ZeroBase  := 2048;
   FullRange := 2048;
   VoltageRange := 10;
   Gain := Gain mod 16;
   If (Gain < 0) Or (Gain > 3) Then
   begin
      P100X_AD2F := -100;
      exit;
   end ;

   If HiLo = 0 Then  //Low-Gain
      Result := ((((hex - ZeroBase) / FullRange) * VoltageRange) / Power(2,
Gain))
   Else
      Result := ((((hex - ZeroBase) / FullRange) * VoltageRange) /
Power(10,Gain));

End;

end.
```

3. Demo Result

3.1 Visual C++

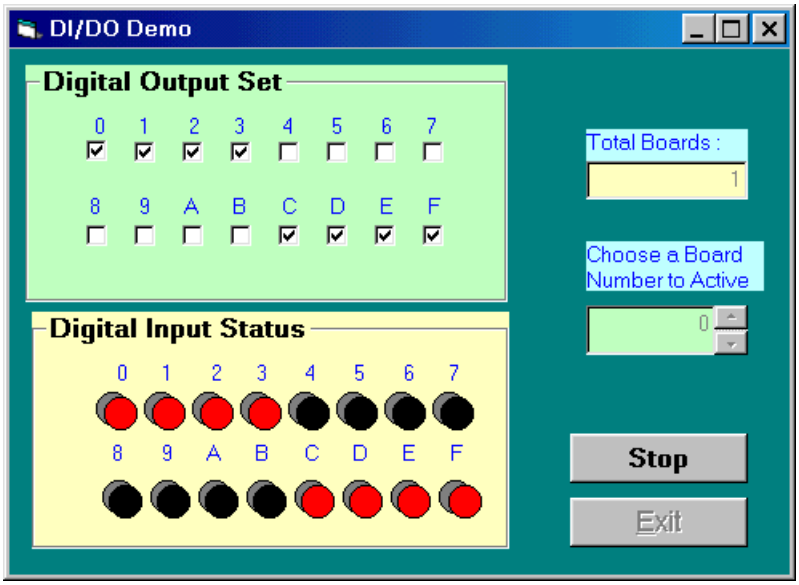


Analog Input with polling demo program

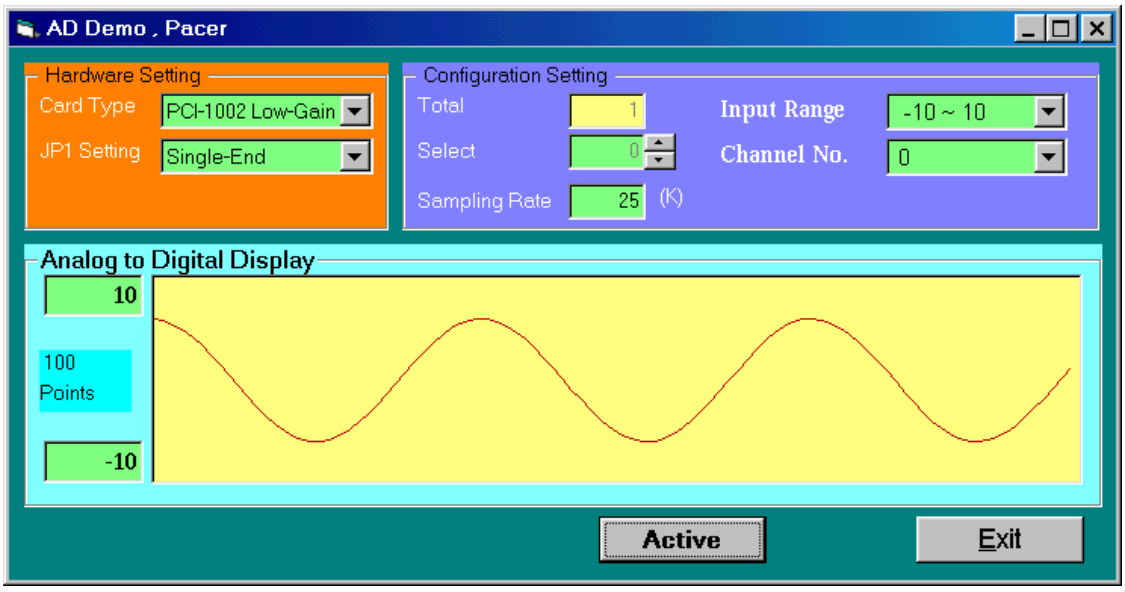


Digital I/O with MFC demo program

3.2 Visual Basic

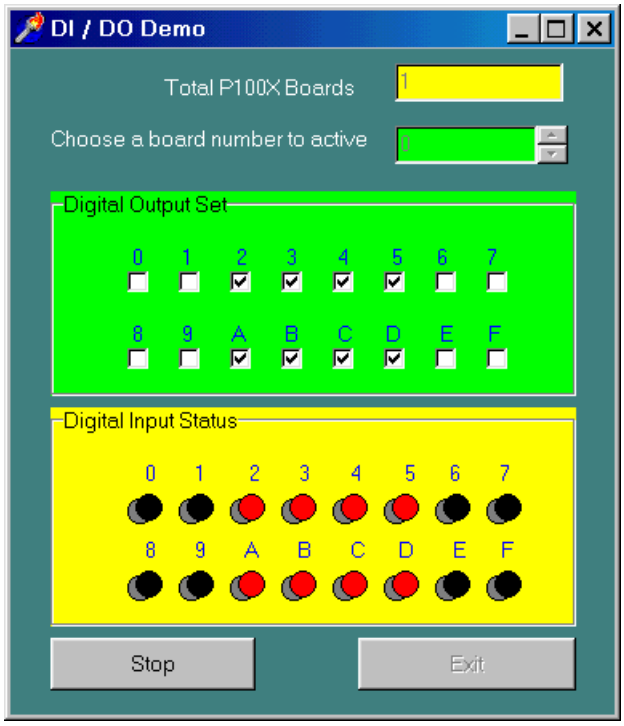

DI/DO Demo
Digital Output Set
0 1 2 3 4 5 6 7
8 9 A B C D E F
Digital Input Status
0 1 2 3 4 5 6 7
8 9 A B C D E F
Total Boards :
1
Choose a Board Number to Active
0
Stop
Exit

Digital I/O demo program

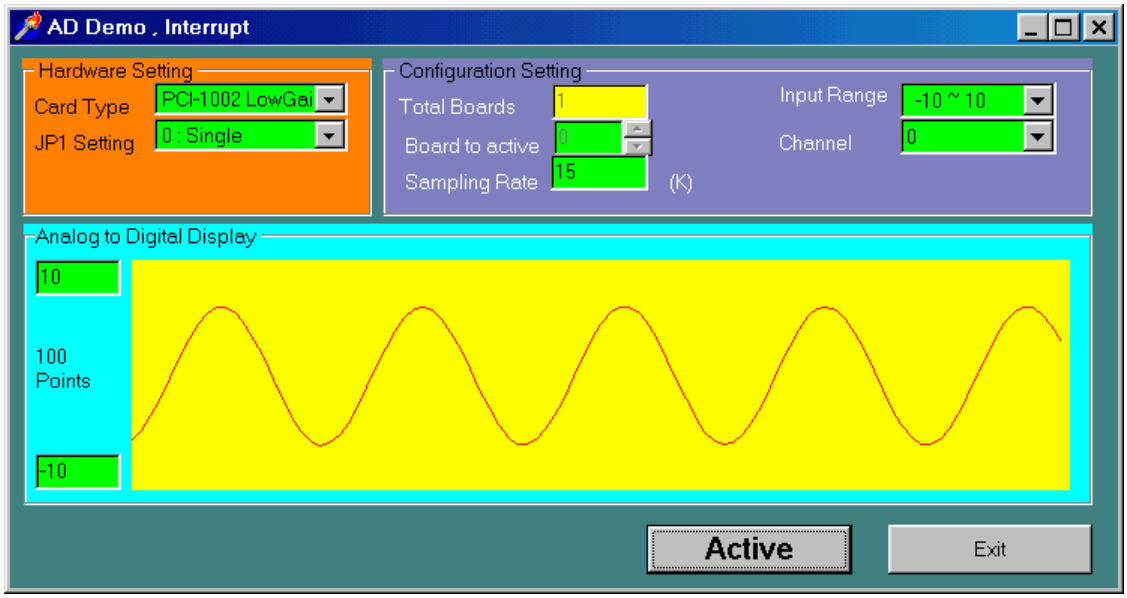


Analog Input with pacer-trigger demo program

3.3 Delphi

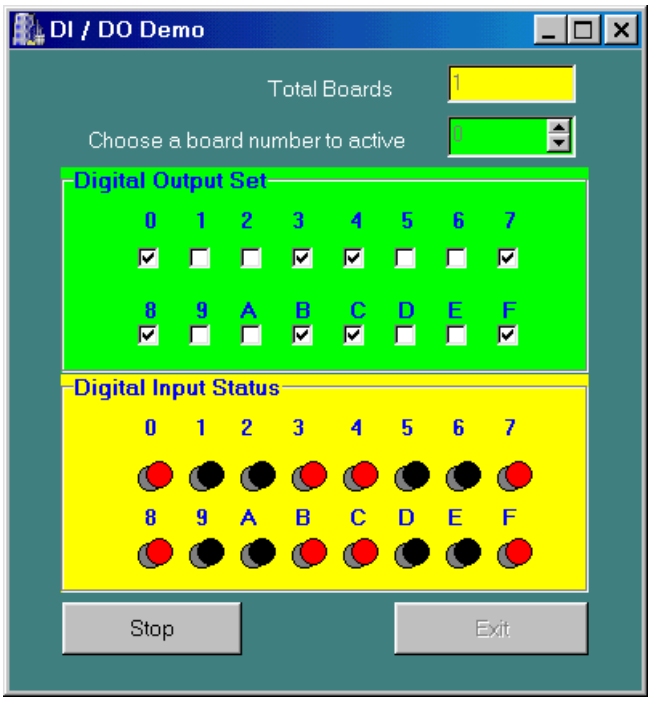

DI / DO Demo
Total P100X Boards
1
Choose a board number to active
0
Digital Output Set
0 1 2 3 4 5 6 7
8 9 A B C D E F
Digital Input Status
0 1 2 3 4 5 6 7
8 9 A B C D E F
Stop
Exit

Digital I/O demo program

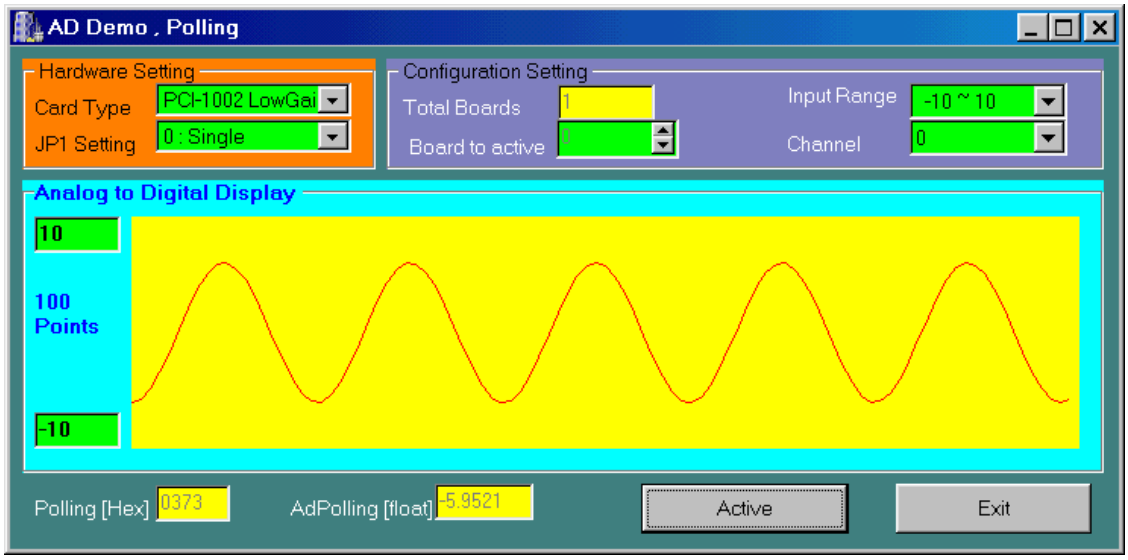


Analog Input with Interrupt demo program

3.4 Borland C++ Builder


DI / DO Demo
Total Boards
1
Choose a board number to active
0
Digital Output Set
0 1 2 3 4 5 6 7
8 9 A B C D E F
Digital Input Status
0 1 2 3 4 5 6 7
8 9 A B C D E F
Stop
Exit

Digital I/O demo program



Analog Input with polling demo program

4. Description of Functions

These function in DLL are divided into several groups as following:

- The test functions
- The D/I/O functions
- The A/D fixed-mode functions
- The Driver functions
- The Interrupt functions

The functions of test listing as follows:

1. P100X_FloatSub2
2. P100X_ShortSub2
3. P100X_GetDllVersion
4. P100X_GetDriverVersion

The functions of DI/O listing as follows:

1. P100X_DI
2. P100X_DO

The functions of fixed-channel mode listing as follows:

1. P100X_SetChannelConfig
2. P100X_Polling
3. P100X_AdPolling
4. P100X_AdsPolling
5. P100X_AdsPacer

The functions of Driver listing as follows:

1. P100X_DriverInit
2. P100X_DriverClose
3. P100X_GetConfigAddressSpace
4. P100X_WhichBoardActive
5. P100X_ActiveBoard
6. P100X_GetIrqNo

The functions of Interrupt listing as follows:

1. P100X_InstallIrq
2. P100X_INT_AdStart
3. P100X_ExtINT_AdStart
4. P100X_INT_AdStop
5. P100X_GetIntCount
6. P100X_GetBuffer
7. P100X_GetFloatBuffer

In this chapter, we use some keywords to indicate the attribute of Parameters.

Keyword	Setting parameter by user before calling this function ?	Get the data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

Note: All of the parameters need to be allocated spaces by the user.

4.1 The Configuration Code Table

PCI-1002L Configuration Code Table

Gain	Bipolar	Max. Switching Frequency	Configuration Code
1	+/- 10V	110 K/S	0x00
2	+/- 5.0V	110 K/S	0x01
4	+/- 2.5V	110 K/S	0x02
8	+/- 1.25V	110 K/S	0x03

PCI-1002H Configuration Code Table

Gain	Bipolar	Max. Switching Frequency	Configuration Code
1	+/- 10V	44 K/S	0x10
10	+/- 1.0V	36 K/S	0x11
100	+/- 0.1V	7 K/S	0x12
1000	+/- 0.01V	0.8 K/S	0x13

4.2 The Test Functions

4.2.1 P100X_FloatSub2

- **Description:**
  Calculates C = fA - fB in **float** format, **float=4 bytes floating point number.** This function is provided to test DLL linkage

- **Syntax:**
  float P100X_FloatSub2(float fA, float fB);

- **Parameter:**
  fA : [Input] 4 bytes floating point value
  fB : [Input] 4 bytes floating point value

- **Return:**
  Return the result value (= fA - fB).

4.2.2 P100X_ShortSub2

- **Description :**

- Calculates C = nA - nB in SHORT formats, **SHORT=16 bits signed number.** This function is provided to test DLL linkage.

- **Syntax :**
  short P100X_ShortSub2(Short nA, Short nB);

- **Parameter:**
  nA : [Input] 16-bit value
  nB : [Input] 16-bit value

- **Return:**
  Return the result value (= nA - nB).

4.2.3 P100X_GetDllVersion

- **Description :**
  Read the DLL version number of the **P100X.DLL**.

- **Syntax :**
  WORD P100X_GetDllVersion(void);

- **Parameter:**
  None

- **Return:**
  Return the version of DLL for Device-Driver.
  return=0x200 → Version 2.0

4.2.4 P100X_GetDriverVersion

- **Description :**
  This subroutine will read the software version number of P100X.VxD of Windows 95 or P100X.SYS of Windows NT.

- **Syntax :**
  WORD P100X_GetDriverVersion(WORD *wDriverVersion);

- **Parameter:**
  wDriverVersion : [Output] address of **wDriverVersion,**
  which will stores the version of Device-Driver.
  wDriverVersion=0x200 → Version 2.0

- **Return:**
  P100X_NoError : OK
  P100X_DriverHandleError : the P100X.VxD open error for Windows 95
  the P100X.SYS open error for Windows NT/2000/XP

  P100X_DriverCallError : call P100X.VxD return error
  call P100X.SYS return error

4.3 The DI/O Functions

4.3.1 P100X_Di

- **Description :**
  This subroutine will read the 16 bits data from DI port. This function will refer to the current active PCI-100X. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_Di(WORD *wDi);

- **Parameter:**
  wDi : [Output] address of **wDi**,
  which will stores the 16 bits data of Digital-Input.

- **Return:**
  P100X_NoError : OK
  P100X_FindBoardError : cannot find the PCI-100X board
  P100X_ExceedBoardNumber : invalidate board number

4.3.2 P100X_Do

- **Description :**
  This subroutine will send the 16 bits data to DO port. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**
  WORD P100X_Do(WORD wDo);

- **Parameter:**
  wDo : [Input] the 16 bit data sent to Digital-Output port

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board

4.4 The AD Fixed-mode Functions

4.4.1 P100X_SetChannelConfig

- **Description :**

  This subroutine will set the AD channel's configuration code. This subroutine will set the active AD channel for **P100X_AdPolling**, **P100X_AdsPolling** and **P100X_AdsPacer**. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_SetChannelConfig(WORD wChannel, WORD wConfig);

- **Parameter:**

  wChannel : [Input] AD channel number
  wConfig : [Input] Configuration code. Refer to Sec. 3.1 for details.

- **Return:**

  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdControllerError : MagicScan controller hardware handshake error

4.4.2 P100X_Polling

- **Description :**

  Performs a single A/D conversion on the active channel by software polling. The **P100X_SetChannelConfig** subroutine can be used to change channel or configuration code. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_Polling(word *wAdVal);

- **Parameter:**

  wAdVal : [Output] address of **wAdVal**, which store the AD data
  Data is returned as an integer value in the range 0-4095.

- **Return:**

  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdPollingTimeOut : hardware timeout error

4.4.3 P100X_AdPolling

- **Description :**

  This subroutine will perform the AD conversion by polling one time. The **P100X_SetChannelConfig** subroutine can be used to change channel or configuration code and the **P100X_AdPolling** will refer to that condition in later operation. This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_AdPolling(float *fAdVal);

- **Parameter:**

  fAdVal : [output] address of **fAdVal**, which will stores the AD data. this data is automatically computed based on the setting of **P100X_SetChannelConfig()**.

- **Return:**

  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdPollingTimeOut : hardware timeout error

4.4.4 P100X_AdsPolling

- **Description :**

  Performs multiple A/D conversions on a single channel by polling. The **P100X_SetChannelConfig** subroutine can be used to change the channel or configuration code. This function addresses the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

  Since software polling can be interrupted by the operating system, the P100X_AdsPacer function is recommended when precisely reconstructing the waveform is desired.

- **Syntax :**

  WORD P100X_AdsPolling(float fAdVal[], WORD wNum);

- **Parameter:**

  fAdVal : [Output] starting address of AD data buffers(Array of float), these data will be automatically computed based on the setting of **P100X_SetChannelConfig()**.

  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

  wNum : [Input] number of AD conversions will be performed.

- **Return:**

  P100X_NoError : OK
  P100X_ExceedBoardNumber : Invalidate board number
  P100X_FindBoardError : Can not find the PCI-1002 board
  P100X_AdPollingTimeOut : Hardware timeout error

4.4.5 P100X_AdsPacer

- **Description :**

  This subroutine will perform multiple AD conversions by pacer trigger. The **P100X_SetChannelConfig** subroutine can be used to change channel or configuration code and the **P100X_AdsPacer** will refer to that condition in later operation. The hardware pacer will generate trigger signal to AD converter periodically. So these AD data can be used to reconstruct the waveform of analog input. Software polling controls the P100X_AdsPolling , so the AD conversion operation will be interrupted by system OS. **It is recommended to use *P100X_AdsPacer* if the input wave form reconstruction is needed.** This function will refer to the current active PCI-1002 board. Use the P100X_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P100X_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);

- **Parameter:**

  fAdVal : [Output] Address of AD data buffers (Array of WORD), these data will be automatically computed based on the setting of **P100X_SetChannelConfig()**.
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

  wNum : [Input] number of AD conversions will be performed.
  wSample : [Input] **AD sampling rate = 4M/wSample.**

- **Return:**

  P100X_NoError : OK
  P100X_ExceedBoardNumber : invalidate board number
  P100X_FindBoardError : cannot find the PCI-1002 board
  P100X_AdPollingTimeOut : hardware timeout error

4.5 The Driver Functions

4.5.1 P100X_DriverInit

- **Description :**
  This function will detect all the PCI-1002 boards installed in the system. This function must be called once before the other functions are called.

- **Syntax :**
  WORD P100X_DriverInit(WORD *wTotalBoard);

- **Parameter:**
  wTotalBoard : [Output] Address of **wTotalBoard,** which will stores the number of PCI-1002 boards on the system.
  wTotalBoard=0 → Not found.
  wTotalBoard=1 → one PCI-1002 card in the system
  wTotalBoard=n → n*PCI-1002 cards in the system

- **Return:**

P100X_NoError	: OK
P100X_NoFoundBoard	: can not detect any PCI-1002
P100X_FindBoardError	: handshake check error
P100X_DriverHandleError	: the P100X.VxD .open error for Windows 95 the P100X.SYS .open error for Windows NT
P100X_DriverCallError	: call P100X.VxD return error call P100X.SYS return error

4.5.2 P100X_DriverClose

- **Description :**
  Release all resources to system. This function must be called once before program is terminated.

- **Syntax :**
  void P100X_DriverClose(void);

- **Parameter:**
  None

- **Return:**
  None

4.5.3 P100X_GetConfigAddressSpace

- **Description :**
  Get the I/O address of PCI-1002 board n. This function is for debug. It is not necessary to call this function.

- **Syntax :**
  WORD P100X_GetConfigAddressSpace(WORD wBoardNo,
  WORD wAddrTimer, WORD *wAddrDio, WORD *wAddrAd);

- **Parameter:**
  wBoardNo : [Input] PCI-1002 board number
  wAddrTimer, wAddrDio, wAddrAd
  : [Output] Address of wAddrTimer, wAddrDio, wAddrAD which will stores the address for access the Timer, DI/DO and AD. Please refer to Hardware manual for details.

- **Return:**
  P100X_NoError : OK
  P100X_FindBoardError : handshake check error
  P100X_ExceedBoardError : wBoardNo is invalidated

4.5.4 P100X_WhichBoardActive

- **Description:**
  Return the board number of the active board.

- **Syntax:**
  WORD P100X_WhichBoardActive(void);

- **Parameter:**
  None

- **Return:**
  Return the board number of the active board.

4.5.5 P100X_ActiveBoard

- **Description:**
  This function wills active one of the PCI-1002 boards installed in the system. This function must call once before the D/I/O, A/D, D/A functions are called.

- **Syntax:**
  WORD P100X_ActiveBoard(WORD wBoardNo);

- **Parameter:**
  wBoardNo : [Input] The board numbers to active.

- **Return:**
  P100X_NoError : OK
  P100X_ExceedBoardError : wBoardNo is invalidated

4.5.6 P100X_GetIrqNo

- **Description:**
  This function will get the IRQ number of the active PCI-1002 board installed in the system. It can let you known what IRQ number is used by PCI-1002. This function is not the necessary for your program.

- **Syntax:**
  WORD P100X_GetIrqNo(WORD *IrqNo);

- **Parameter:**
  IrqNo : [Output] Address of IrqNo, which will stores the IRQ No that allocated by the system.

- **Return:**
  P100X_NoError : OK

4.6 The Interrupt Functions

4.6.1 P100X_InstallIrq

- **Description :**
  This subroutine will install interrupt handler for a specific IRQ n. and setting the maximum number of interrupts. Refer to Section 3.6.7. for more details on using interrupts.

- **Syntax :**
  WORD P100X_InstallIrq(HANDLE *hEvent, DWORD dwCount);

- **Parameter:**
  hEvent : [Input] The user must uses the CreateEvent() to create the Event object and obtain its handle and pass the handle into this function.
  dwCount : [Input] Maximum numbers of counter for interrupt transfer.

- **Return:**
  P100X_NoError : successful
  P100X_InstallIrqError : fail in install IRQ handler.

4.6.2 P100X_GetIntCount

- **Description :**
This subroutine will read the interrupt transfer count.
- **Syntax :**
  WORD P100X_GetIntCount(DWORD *dwVal)

- **Parameter:**
  dwVal : [Output] the address of dwVal, which will store the value of interrupt transferred count.

- **Return:**
  P100X_NoError : successful
  P100X_GetIntCountError : fail get interrupt count.

4.6.3 P100X_INT_AdStart

- **Description :**
  This subroutine will start the interrupt transfer for a specific A/D channel and programming the gain code and sampling rate.

- **Syntax :**
  WORD P100X_INT_AdStart(WORD Ch, WORD Gain, WORD wFreqDiv)

- **Parameter:**
  Ch : [Input] the A/D channel.
  Gain : [Input] the Gain, refer to Section 3.1
  wFreqDiv : [Input] the sampling rate is 4M/(wFreqDiv)

- **Return:**
  P100X_NoError : successful
  P100X_INTStartError : failure

4.6.4 P100X_ExtInt_AdStart

- **Description :**
  This subroutine will start the interrupt transfer for a specific A/D channel and programming the gain code from external interrupt (Pin19).

- **Syntax :**
  WORD P100X_ExtINT_AdStart(WORD Ch, WORD Gain)

- **Parameter:**
  Ch : [Input] the A/D channel.
  Gain : [Input] the Gain, refer to Section 3.1

- **Return:**
  P100X_NoError : successful
  P100X_INTStartError : failure

4.6.5 P100X_INT_AdStop

- **Description :**
  This subroutine will stop the interrupt transfer and remove the installed interrupt handler.

- **Syntax :**
  WORD P100X_ INT_AdStop(void)

- **Parameter:**
  None

- **Return:**
  P100X_NoError : successful
  P100X_INTStopError : failure

4.6.6 P100X_GetBuffer

- **Description :**
  This subroutine will copy the transferred interrupted data into the user's buffer (in word format).

- **Syntax :**
  WORD P100X_GetBuffer(DWORD dwNum, WORD wBuffer[])

- **Parameter:**
  wNum : [Input] The total number to transfer to User's Buffer.
  wBuffer : [Output] The address of wBuffer (Array of word) that store the Hex A/D value.
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

- **Return:**
  P100X_NoError : successful
  P100X_GetBufferError : fail

4.6.7 P100X_GetFloatBuffer

- **Description :**
  This subroutine will copy the transferred interrupted data into the user's buffer (in floating-point format).

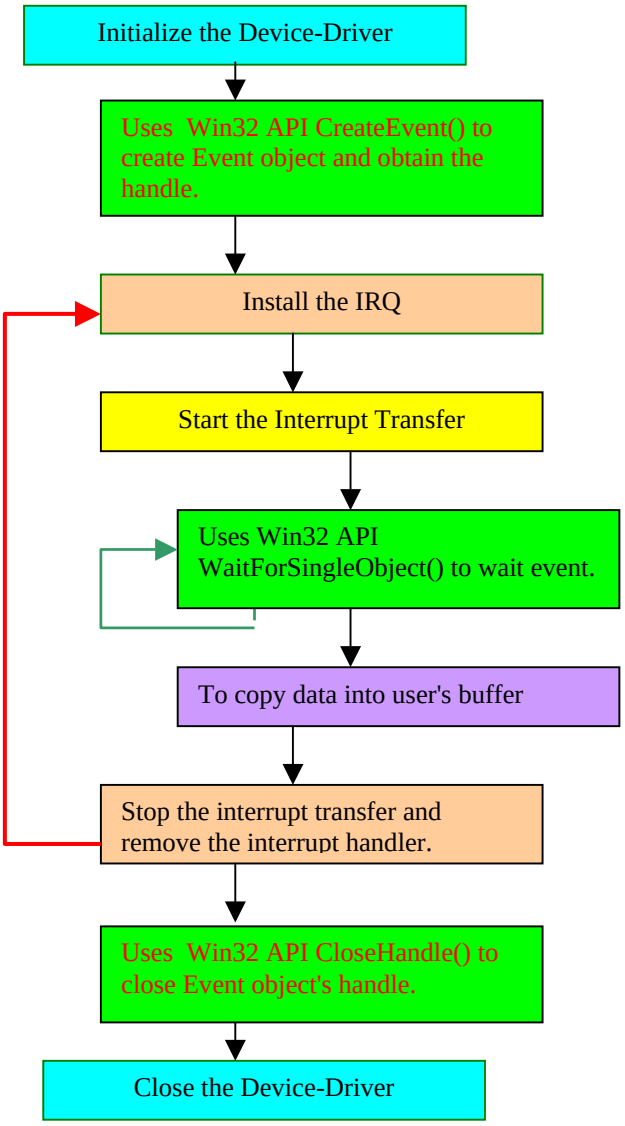
- **Syntax :**
  WORD P100X_GetFloatBuffer(DWORD dwNum, float fAdVal[])

- **Parameter:**
  wNum : [Input] The total number to transfer to User's Buffer.
  fAdVal : [Output] Address of fAdVals (Array of float) that store the voltage value(floating-point).
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user cans analyze these data from the buffer after calling this function.

- **Return:**
  P100X_NoError : successful
  P100X_GetBufferError : fail

4.6.8 Architecture of Interrupt mode

The 3.6.1 to 3.6.6 are these functions to perform the A/D conversion with interrupt transfer. The flow chart to program these functions is given as follows:


Initialize the Device-Driver
Uses Win32 API CreateEvent() to create Event object and obtain the handle.
Install the IRQ
Start the Interrupt Transfer
Uses Win32 API WaitForSingleObject() to wait event.
To copy data into user's buffer
Stop the interrupt transfer and remove the interrupt handler.
Uses Win32 API CloseHandle() to close Event object's handle.
Close the Device-Driver

P100X_DriverInit( )

…….

CreateEvent( )

…….

P100X _InstallIrq(…)

……

P100X _INT_AdStart(…)

……

WaitForSingleObject(….)

……..

P100X _GetBuffer(….)

……

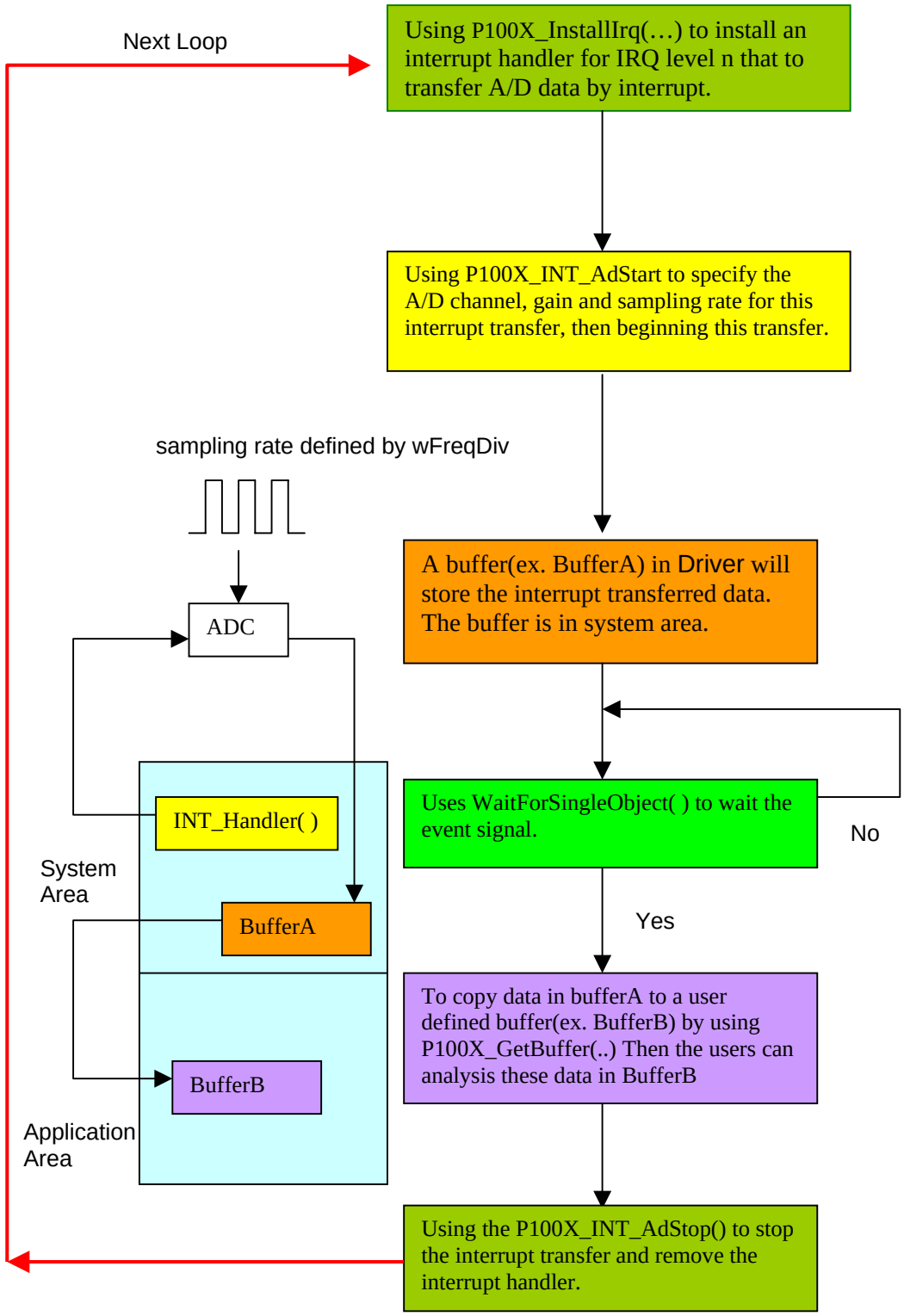
P100X _INT_AdStop

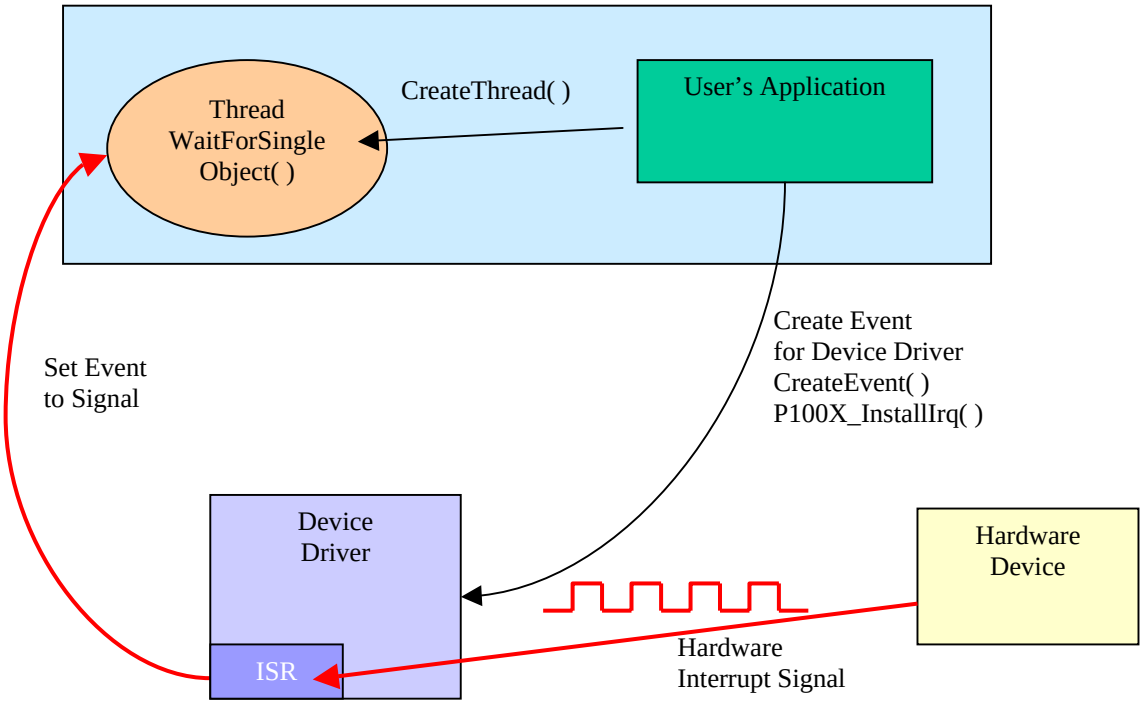
……

CloseHandle( )

…….

P100X _DriverClose( )


Next Loop
Using P100X_InstallIrq(…) to install an interrupt handler for IRQ level n that to transfer A/D data by interrupt.
Using P100X_INT_AdStart to specify the A/D channel, gain and sampling rate for this interrupt transfer, then beginning this transfer.
sampling rate defined by wFreqDiv
A buffer(ex. BufferA) in Driver will store the interrupt transferred data. The buffer is in system area.
ADC
INT_Handler( )
System Area
BufferA
Uses WaitForSingleObject( ) to wait the event signal.
No
Yes
To copy data in bufferA to a user defined buffer(ex. BufferB) by using P100X_GetBuffer(..) Then the users can analysis these data in BufferB
BufferB
Application Area
Using the P100X_INT_AdStop() to stop the interrupt transfer and remove the interrupt handler.



Please refer to the following Windows API functions:

The following description of these functions was copied from MSDN. Refer to MSDN for complete details.

CreateEvent( )

The CreateEvent function creates or opens a named or unnamed event object.

```
HANDLE CreateEvent(
    // pointer to security attributes
    LPSECURITY_ATTRIBUTES lpEventAttributes,
    BOOL bManualReset,      // flag for manual-reset event
    BOOL bInitialState,         // flag for initial state
    LPCTSTR lpName          // pointer to event-object name
);
```

CreateThread( )

The CreateThread function creates a thread to execute within the virtual address space of the calling process.

To create a thread that runs in the virtual address space of another process, use the CreateRemoteThread function.

```
HANDLE CreateThread(
    // pointer to security attributes
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    DWORD dwStackSize,              // initial thread stack size
     // pointer to thread function
    LPTHREAD_START_ROUTINE lpStartAddress,
    LPVOID lpParameter,             // argument for new thread
    DWORD dwCreationFlags,          // creation flags
    LPDWORD lpThreadId              // pointer to receive thread ID
);
```

WaitForSingleObject( )

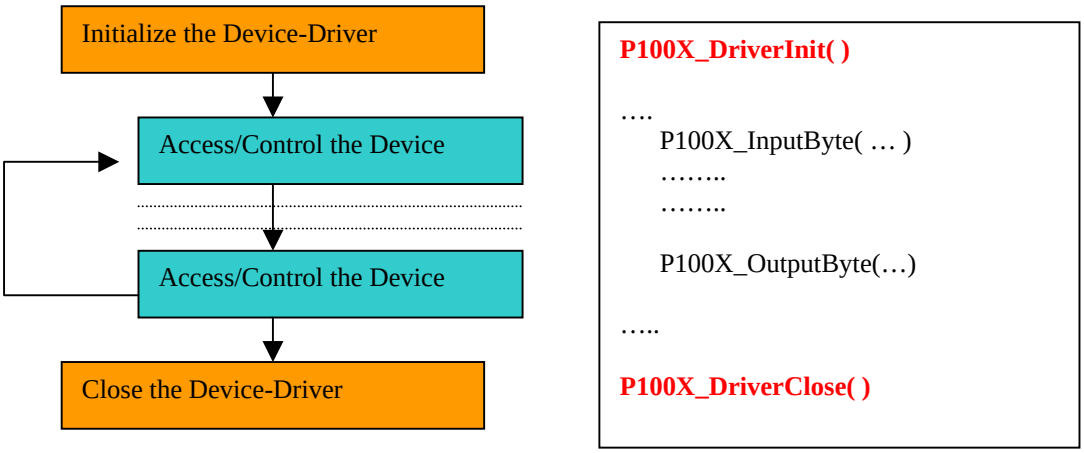
The WaitForSingleObject function returns when one of the following occurs:

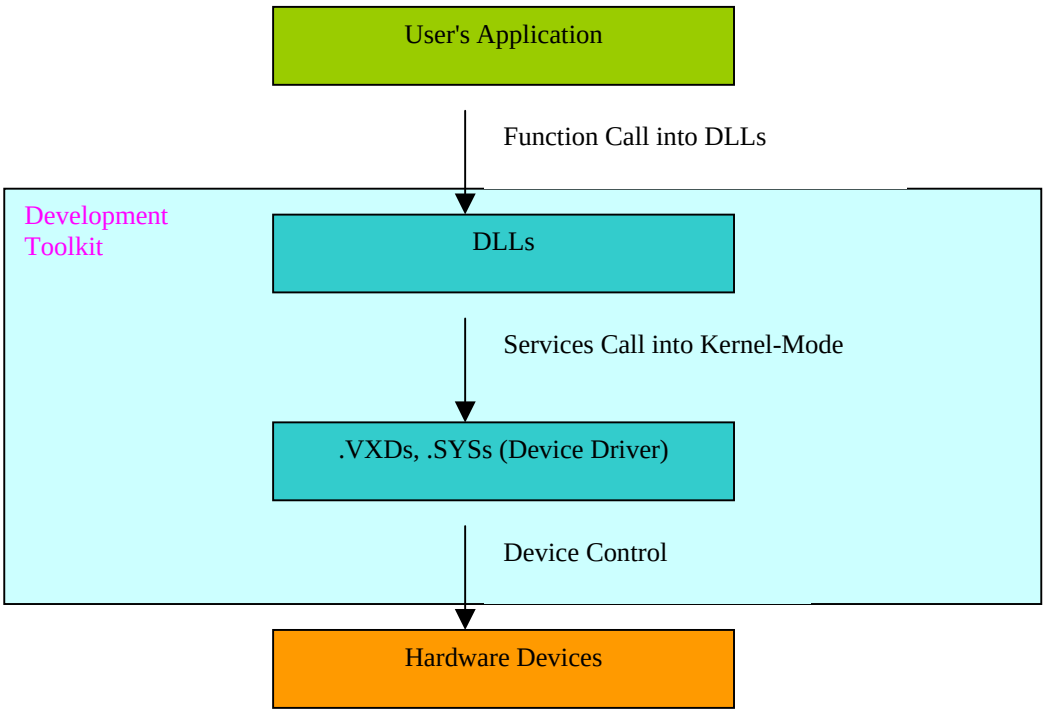
- The specified object is in the signaled state.
- The time-out interval elapses.

To enter an alertable wait state, use the WaitForSingleObjectEx function. To wait for multiple objects, use the WaitForMultipleObjects.

```
DWORD WaitForSingleObject(
        HANDLE hHandle,             // handle to object to wait for
        DWORD dwMilliseconds   // time-out interval in
    milliseconds
);
```

5. Program Architecture





6. Problems Report

Technical support is available at no charge as described below. The best way to report problems is to send electronic mail to

Service@icpdas.com
or icpdas@ms8.hinet.net

on the Internet.

When reporting problems, please include the following information:

1) Is the problem reproducible? If so, how?

2) What kind and version of Platform that you using? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.

3) What kinds of our products that you using? Please see the product's manual .

4) If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.

5) If the problem involves other programs or hardware devices, what devices or version of the failing programs that you using?

6) Other comments relative to this problem or any suggestions will be welcomed.

After we had received your comments, we will take about two business days to testing the problems that you said. And then reply as soon as possible to you. Please check that we had received your comments? And please keeps contact with us.

E-mail: Service@icpdas.com
Web-Site: http://www.icpdas.com